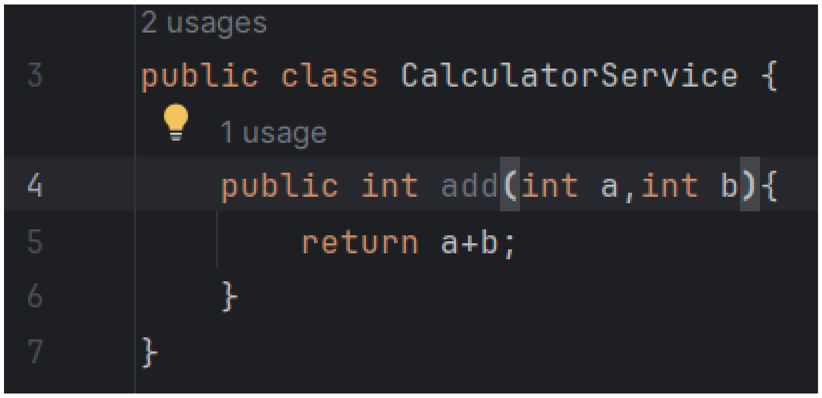
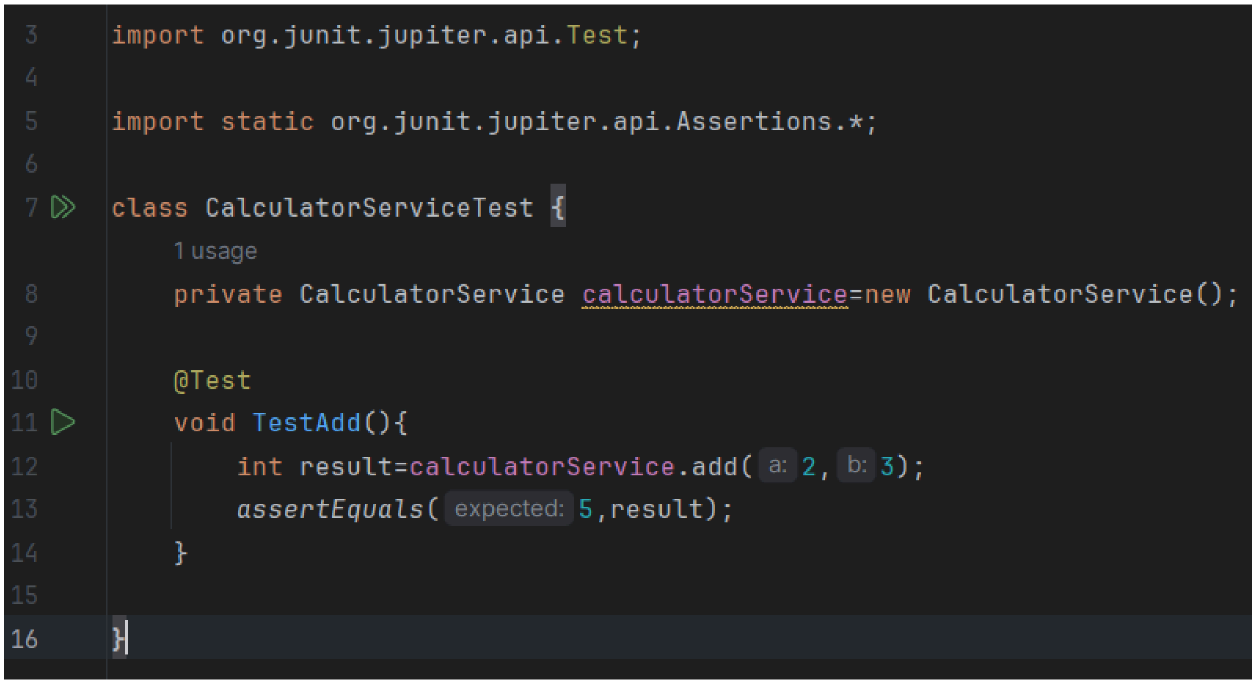
Spring Testing Exercises

Exercise 1: Basic Unit Test for a Service Method

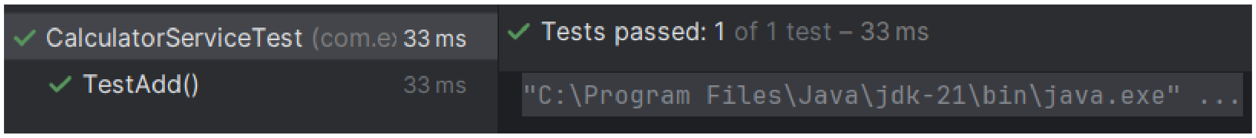
Task: Write a unit test for a service method that adds two numbers.



```
2 usages
public class CalculatorService {
    1 usage
    public int add(int a,int b){
        return a+b;
    }
}
```



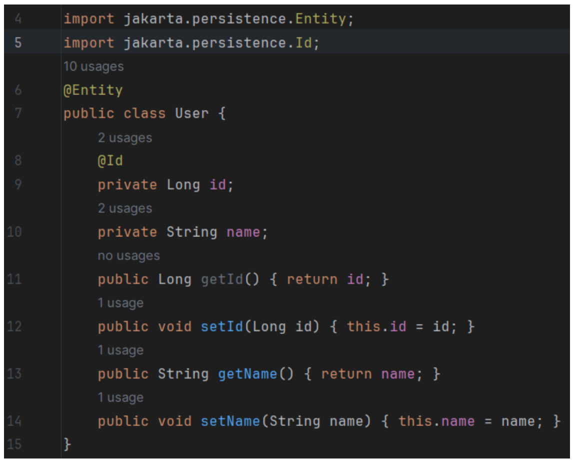
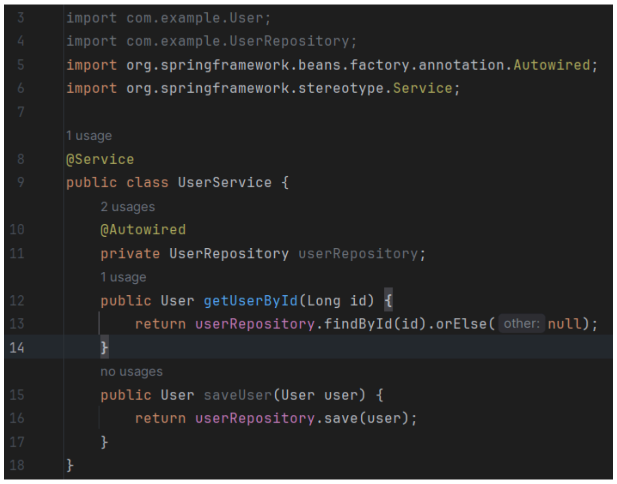
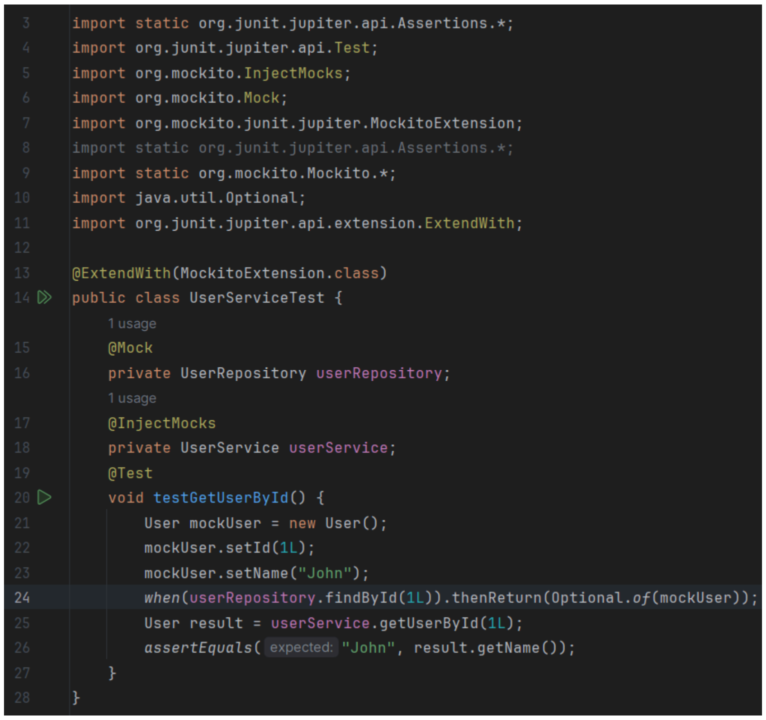
```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class CalculatorServiceTest {
    1 usage
    private CalculatorService calculatorService=new CalculatorService();

    @Test
    void TestAdd(){
        int result=calculatorService.add( a: 2, b: 3);
        assertEquals( expected: 5,result);
    }

}
```

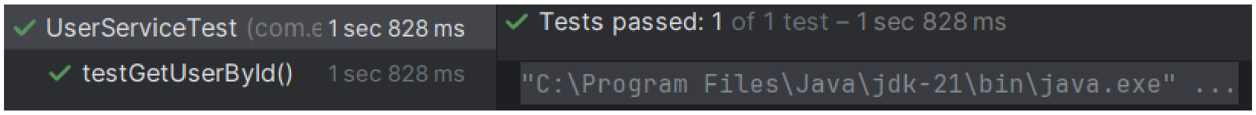


```
✓ CalculatorServiceTest (com.e› 33 ms    ✓ Tests passed: 1 of 1 test – 33 ms
    ✓ TestAdd()                33 ms     "C:\Program Files\Java\jdk-21\bin\java.exe" ...
```

Exercise 2: Mocking a Repository in a Service Test

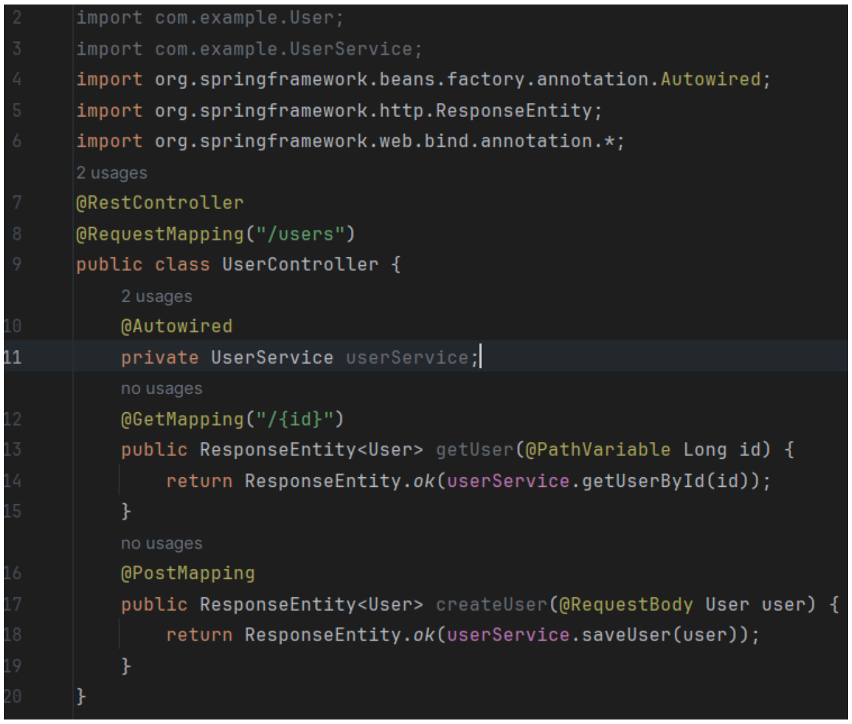
Task: Test a service that uses a repository to fetch data.

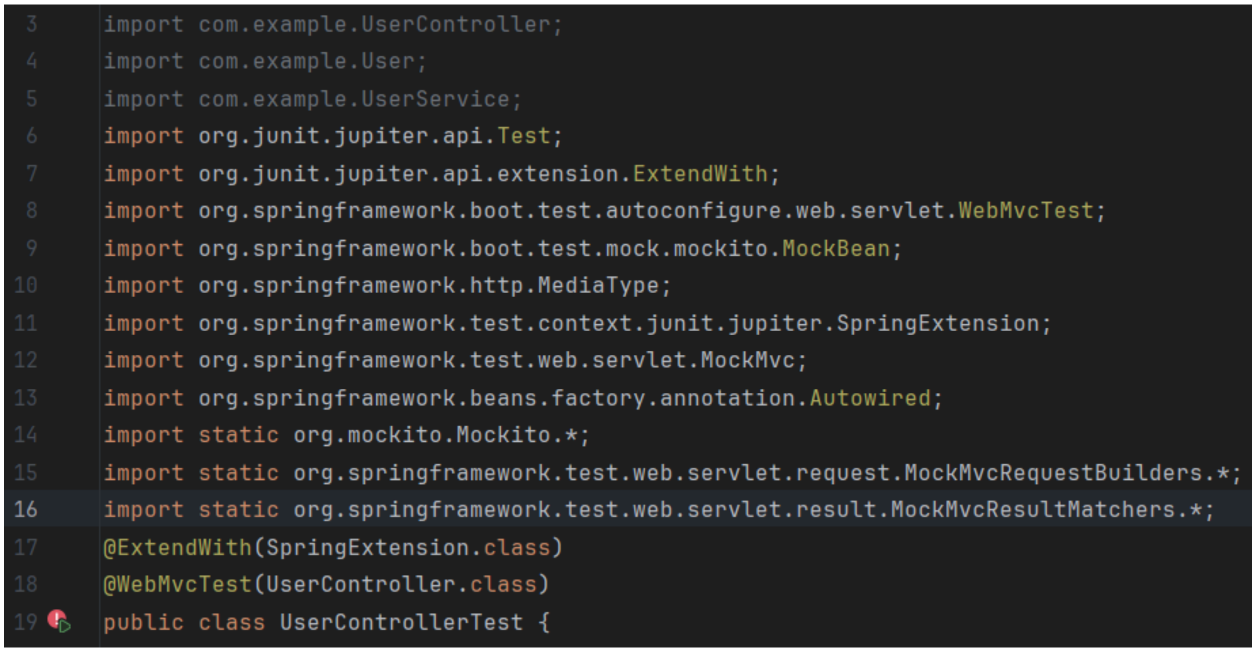

```java
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
10 usages
@Entity
public class User {
    2 usages
    @Id
    private Long id;
    2 usages
    private String name;
    no usages
    public Long getId() { return id; }
    1 usage
    public void setId(Long id) { this.id = id; }
    1 usage
    public String getName() { return name; }
    1 usage
    public void setName(String name) { this.name = name; }
}
```

```java
import com.example.User;
import com.example.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

1 usage
@Service
public class UserService {
    2 usages
    @Autowired
    private UserRepository userRepository;
    1 usage
    public User getUserById(Long id) {
        return userRepository.findById(id).orElse( other: null);
    }
    no usages
    public User saveUser(User user) {
        return userRepository.save(user);
    }
}
```

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;
import java.util.Optional;
import org.junit.jupiter.api.extension.ExtendWith;

@ExtendWith(MockitoExtension.class)
public class UserServiceTest {
    1 usage
    @Mock
    private UserRepository userRepository;
    1 usage
    @InjectMocks
    private UserService userService;
    @Test
    void testGetUserById() {
        User mockUser = new User();
        mockUser.setId(1L);
        mockUser.setName("John");
        when(userRepository.findById(1L)).thenReturn(Optional.of(mockUser));
        User result = userService.getUserById(1L);
        assertEquals( expected: "John", result.getName());
    }
}
```

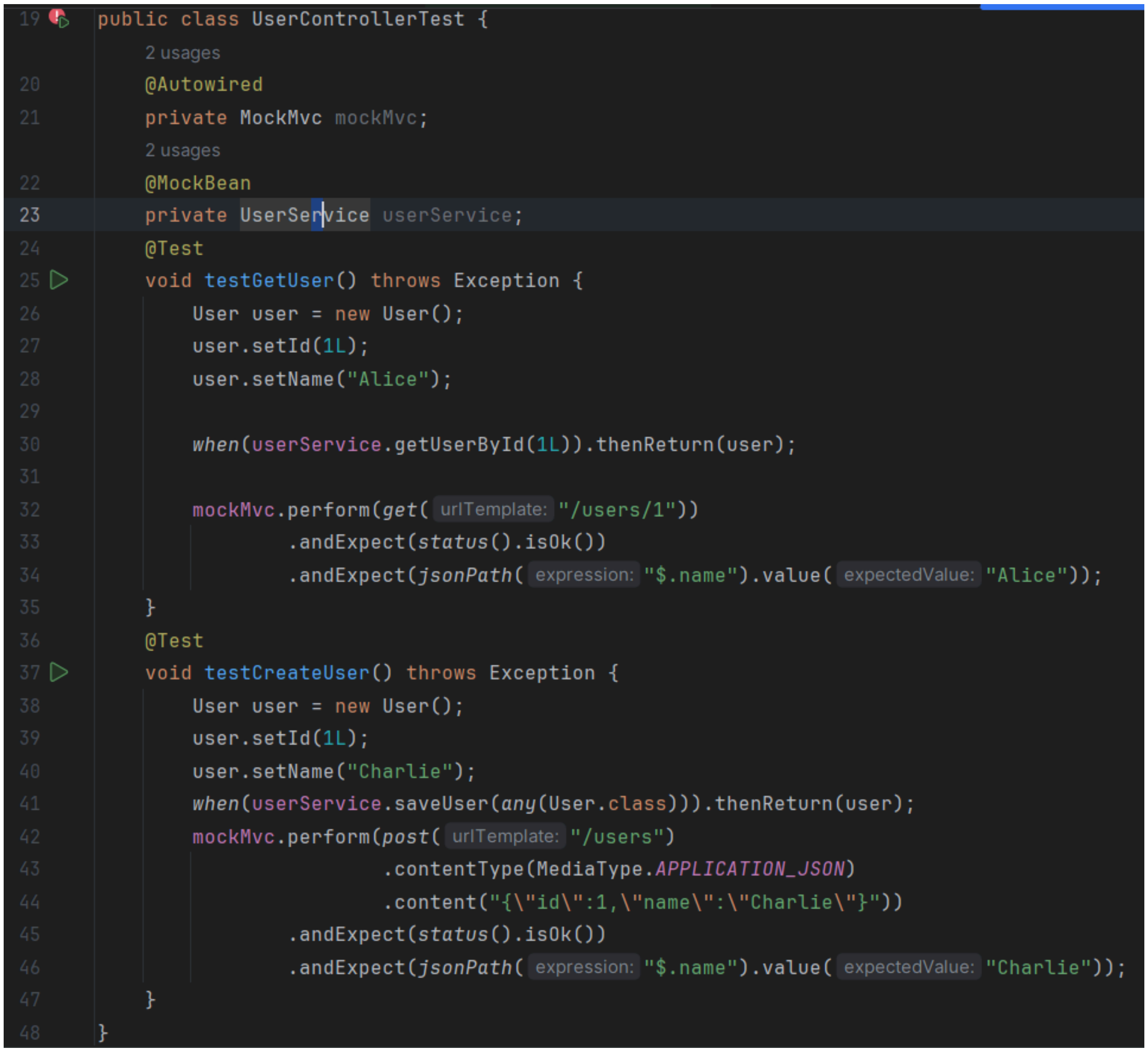

```
✓ UserServiceTest (com.e 1 sec 828 ms      ✓ Tests passed: 1 of 1 test – 1 sec 828 ms
    ✓ testGetUserById()   1 sec 828 ms      "C:\Program Files\Java\jdk-21\bin\java.exe" ...
```

Exercise 3: Testing a REST Controller with MockMvc

Task: Test a controller endpoint that returns a user.


```java
import com.example.User;
import com.example.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
2 usages
@RestController
@RequestMapping("/users")
public class UserController {
    2 usages
    @Autowired
    private UserService userService;
    no usages
    @GetMapping("/{id}")
    public ResponseEntity<User> getUser(@PathVariable Long id) {
        return ResponseEntity.ok(userService.getUserById(id));
    }
    no usages
    @PostMapping
    public ResponseEntity<User> createUser(@RequestBody User user) {
        return ResponseEntity.ok(userService.saveUser(user));
    }
}
```

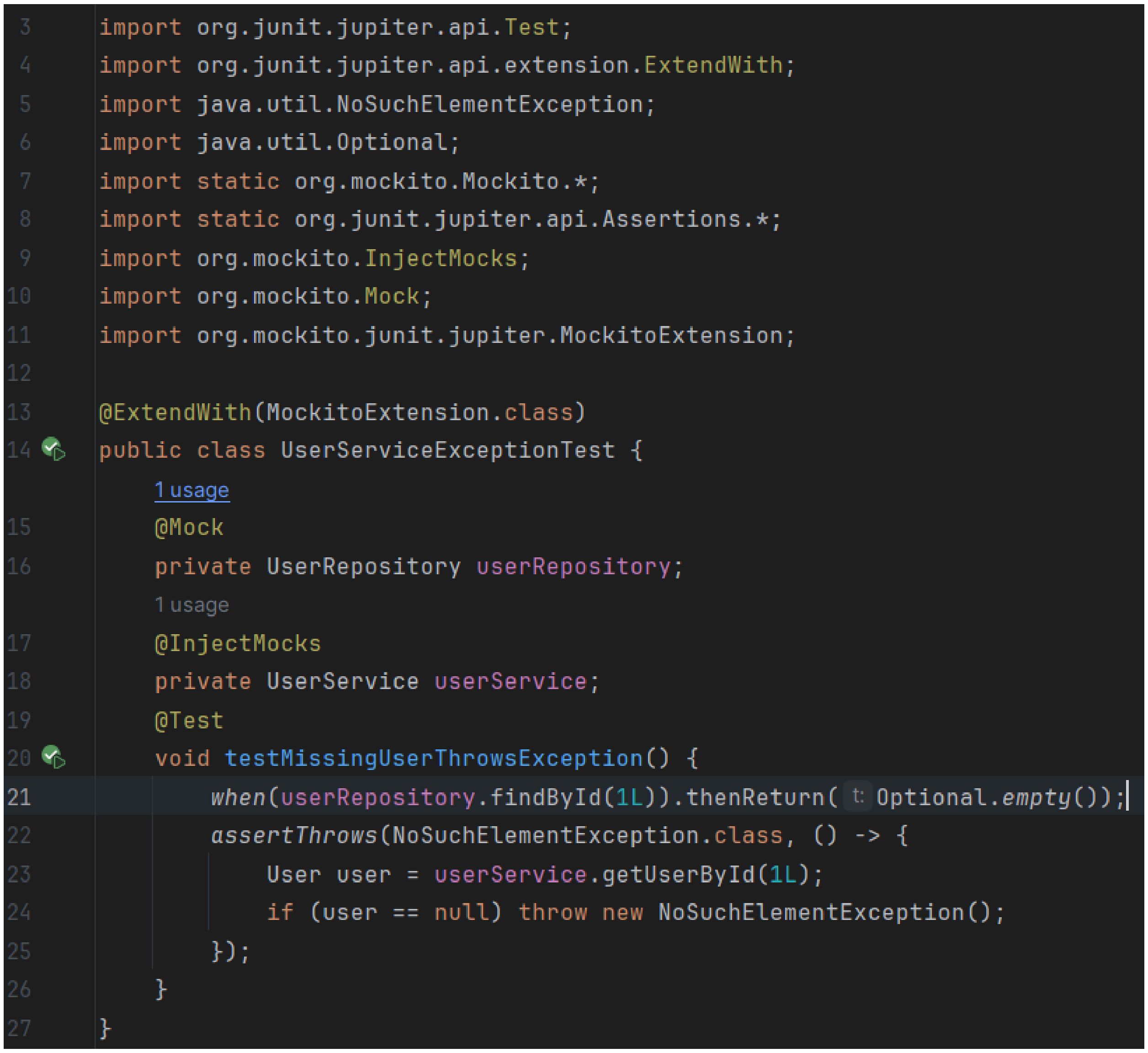

```java
import com.example.UserController;
import com.example.User;
import com.example.UserService;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.http.MediaType;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.beans.factory.annotation.Autowired;
import static org.mockito.Mockito.*;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
@ExtendWith(SpringExtension.class)
@WebMvcTest(UserController.class)
public class UserControllerTest {
```



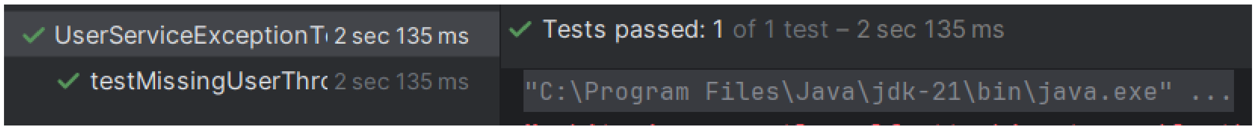
```java
public class UserControllerTest {
    2 usages
    @Autowired
    private MockMvc mockMvc;
    2 usages
    @MockBean
    private UserService userService;
    @Test
    void testGetUser() throws Exception {
        User user = new User();
        user.setId(1L);
        user.setName("Alice");

        when(userService.getUserById(1L)).thenReturn(user);

        mockMvc.perform(get( urlTemplate: "/users/1"))
                .andExpect(status().isOk())
                .andExpect(jsonPath( expression: "$.name").value( expectedValue: "Alice"));
    }
    @Test
    void testCreateUser() throws Exception {
        User user = new User();
        user.setId(1L);
        user.setName("Charlie");
        when(userService.saveUser(any(User.class))).thenReturn(user);
        mockMvc.perform(post( urlTemplate: "/users")
                        .contentType(MediaType.APPLICATION_JSON)
                        .content("{\"id\":1,\"name\":\"Charlie\"}"))
                .andExpect(status().isOk())
                .andExpect(jsonPath( expression: "$.name").value( expectedValue: "Charlie"));
    }
}
```

Exercise 6: Test Service Exception Handling

Task: Test how a service handles a missing user.



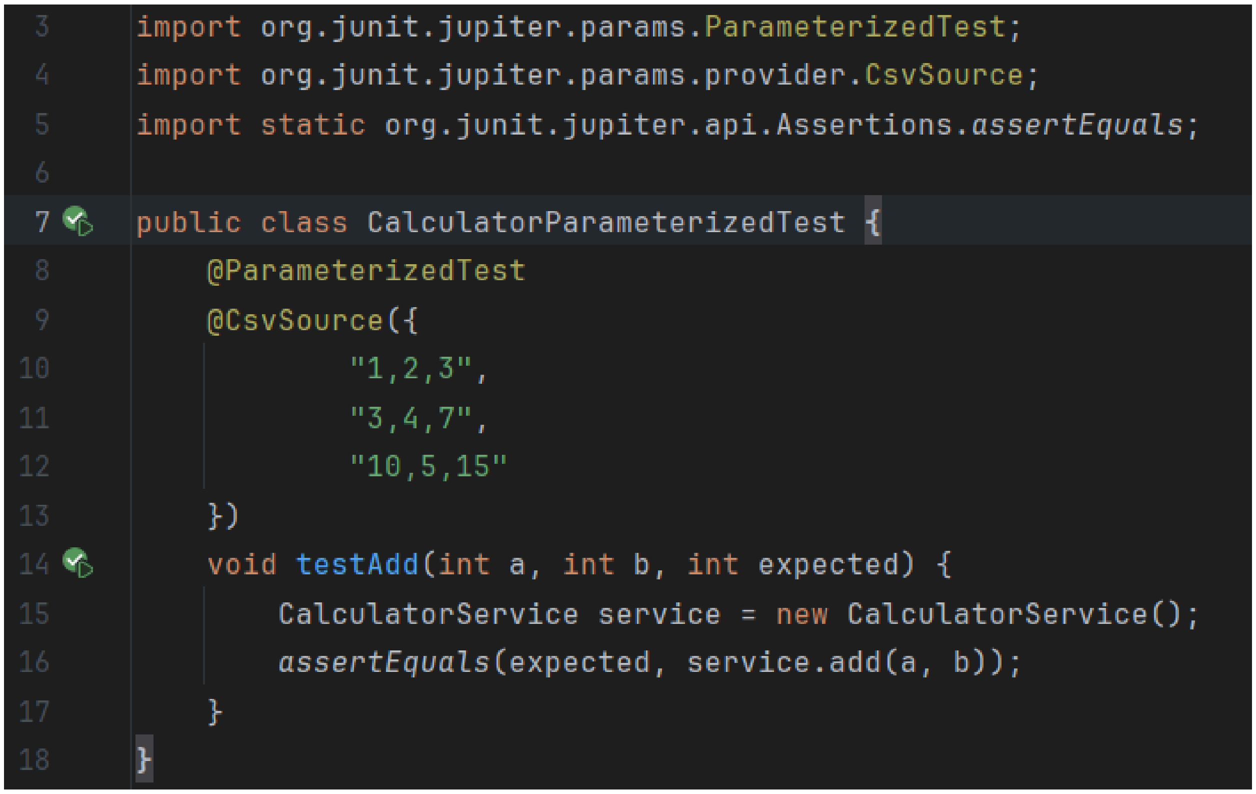
```java
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import java.util.NoSuchElementException;
import java.util.Optional;
import static org.mockito.Mockito.*;
import static org.junit.jupiter.api.Assertions.*;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;

@ExtendWith(MockitoExtension.class)
public class UserServiceExceptionTest {
    1 usage
    @Mock
    private UserRepository userRepository;
    1 usage
    @InjectMocks
    private UserService userService;
    @Test
    void testMissingUserThrowsException() {
        when(userRepository.findById(1L)).thenReturn( t: Optional.empty());
        assertThrows(NoSuchElementException.class, () -> {
            User user = userService.getUserById(1L);
            if (user == null) throw new NoSuchElementException();
        });
    }
}
```



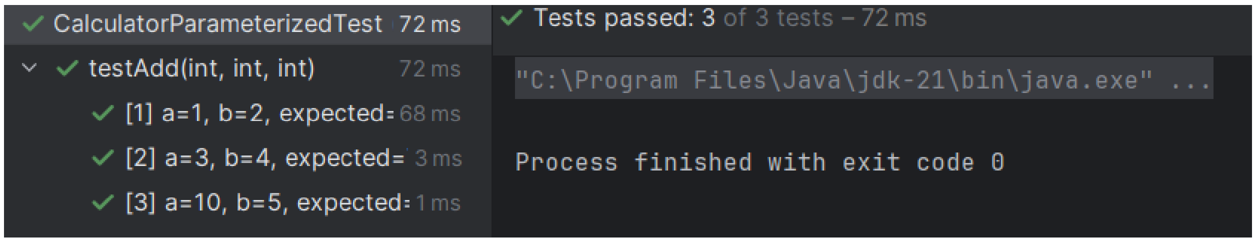
```
✓ UserServiceExceptionT 2 sec 135 ms
  ✓ testMissingUserThrc 2 sec 135 ms
✓ Tests passed: 1 of 1 test – 2 sec 135 ms
"C:\Program Files\Java\jdk-21\bin\java.exe" ...
```

Exercise 9: Parameterized Test with JUnit

Task: Use @ParameterizedTest to test multiple inputs.



```java
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvSource;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class CalculatorParameterizedTest {
    @ParameterizedTest
    @CsvSource({
            "1,2,3",
            "3,4,7",
            "10,5,15"
    })
    void testAdd(int a, int b, int expected) {
        CalculatorService service = new CalculatorService();
        assertEquals(expected, service.add(a, b));
    }
}
```



```
✓ CalculatorParameterizedTest 72 ms
  ✓ testAdd(int, int, int) 72 ms
    ✓ [1] a=1, b=2, expected= 68 ms
    ✓ [2] a=3, b=4, expected= 3 ms
    ✓ [3] a=10, b=5, expected= 1 ms

✓ Tests passed: 3 of 3 tests – 72 ms

"C:\Program Files\Java\jdk-21\bin\java.exe" ...

Process finished with exit code 0
```

I attempted to complete the remaining exercises; however, I encountered numerous errors during execution, which prevented me from finishing them successfully.